April 1981

Hardware and Software Download Techniques With 2816

John F. Rizzo and Randy Battat
Special Products Division
Applications Engineering

INTRODUCTION

Software Updates—how many times in microprocessor systems does software undergo revision? Unfortunately, many people say that it changes frequently. As we all know, such revision can be inconvenient, difficult and extremely costly. The 2816, E^2PROM, from Intel, can not only eliminate these expenses, but increase the functionality of your designs as well. The 2816 combines the benefits of ROM-like non-volatility with RAM-like flexibility. This application notes discusses the costliness of in-field software updates, how 2816 can solve these problems, and some circuit design information detailing how to implement an evolutionary system that eliminates current service costs.

IN FIELD SOFTWARE UPDATES

As technology progresses, the cost of microprocessor systems will become more dependent on design and service costs rather than component costs. Service costs today average about $100/hr. By 1985, assuming a typical inflation rate, those costs will approach $200/hr. Any necessary maintenance to change software, or adjust non-volatile parameters, adds hundreds of dollars to a typical system cost.

To take a realistic example, let's assume a typical microprocessor system (2000 in the field), with a service time of 2 hours per system. Also assume that each system needs to be updated a minimum of 2 times during the product's life. Given such assumptions, the cost involved is at least $400 per system. That's $800,000 for the total retrofit! If one assumes a doubling of labor rates in the next 5 years, the new retrofit cost would be $1.6 million. The 2816 can completely eliminate those costs.

By installing a remote software serial link, the software update can occur over telephone lines, free from service intervention. By 1985 service costs additional to each system will be as much as $800. Adding 2816 and a remote link to the system will cost about $50, a mere one-sixteenth the service cost. Today, a 40% savings can result. Figure 0 shows these cost trends.

It is clear that 2816 can save millions of dollars in maintenance costs. That is why it is such a cost effective solution to the many firmware update problems we face today.

In this application note, the hardware and software designs for such a solution will be discussed. First, though, let us examine the design criteria that are pertinent to the memory elements in such a system:

1) **NON-VOLATILITY**—data must be retained even when the host system is powered down.

2) **FAST ACCESS TIME**—With today's high speed microprocessor systems (i.e., the Intel 8086-2, the Zilog Z8000, and the Motorola MC68000) full throughput is only achieved with fast memory devices. For example, a high performance 8086-2 system for zero wait state operation requires a read access time of 250 ns.

3) **HIGH DENSITY**—As software costs rise, high-level languages will be used to reduce design time. Such high-level languages are often memory intensive, requiring high density memory chips to effectively contain dedicated system programs without sacrificing printed circuit board space.

4) **READ MOSTLY OPERATION**—Program memory and certain types of data memory are mostly accessed in a read mode. There are situations, however, where it is necessary to re-load an entire program (as in the case of a software revision), or reconfigure portions of data storage (e.g., when only certain parameters need to be changed). In these cases, the ability to write to the memory in-circuit is essential.

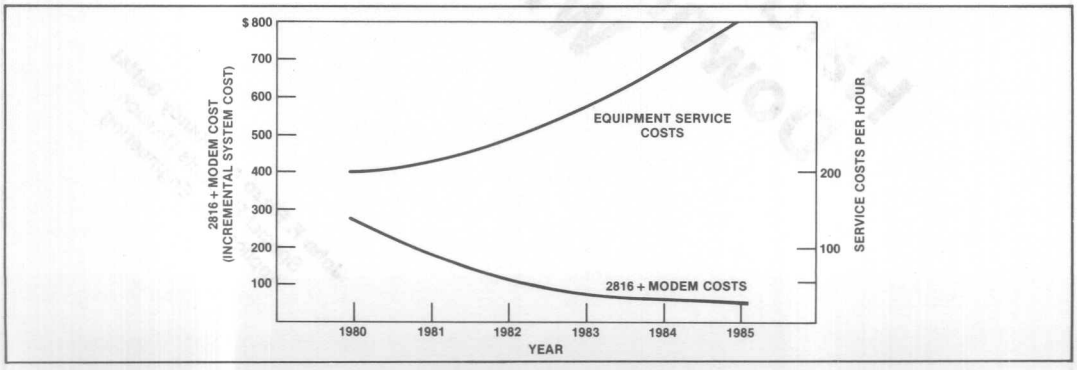


Figure 0. Service Cost Trends

The Intel 2816 fills the need for all these user requirements. It is truly non-volatile, offering greater than 20 year data retention. Access time is 250 ns, which is compatible with today's high speed microcomputer systems. The 2816 is electrically erasable on a per byte or per chip basis—a true read mostly memory, and it offers users 16,384 bits of storage organized as 2048 8-bit bytes.

Specific topics included in this Application Note are the philosophy behind downline loading, as well as the wide spectrum of application possibilities. Included here are four configuration examples. A discussion of both receiving and transmitting functions follows the examples.

DOWNLINE LOAD PHILOSOPHY

The E^2PROM is an excellent medium for storing non-volatile program and data information. The fact that it allows in-circuit erase and write suggests many possibilities as to the information source that the 2816 can be written from. In many instances, E^2PROM memories will be written from remote data facilities.

The telephone is an ideal means of transferring such information, since it is readily available and requires no special interface. With use of an acoustic coupler, serial binary data is converted into high and low frequency tones, which can be transmitted over a datacom link world-wide. Modems interface easily with microprocessors, and the software overhead of performing a downline load operation is minimal.

2816 REMOTE CONFIGURATION OPTIONS

Programs downline loaded to E^2PROMs find many applications in both large and small microcomputer systems. All configurations require a modem to interface electrical signals from a central processor with the acoustically driven telephone. Automatic modems are usually dedicated to a specific telephone line and are completely operated by a host processor. Manual modems are usually portable, relying on the human operator to physically place a telephone receiver in an acoustic coupler cradle, thereby closing the communication loop. Both automatic and manual modems can be used in E^2PROM-telephone communication systems, resulting in four possible configurations:

Manual Receiver — Manual Transmitter

This is a cost effective solution when telephone transmission is not performed often enough to warrant a dedicated telephone line and microprocessor system. Applications include infrequent field updates of program store, where a field system user would call a central factory to have 2816 memory devices reloaded.

Manual Receiver — Automatic Transmitter

Here an automatic transmitter is connected to a microprocessor system which answers the phone and transmits information to 2816s located in remote areas. Applications include field updates, as previously discussed, though a human operator on the transmitting end is not needed. This is advantageous when many field systems will be calling the central factory.

Automatic Receiver — Manual Transmitter

In this situation a microcomputer system would automatically answer the phone to receive information which will eventually be loaded in E^2 devices. This configuration could be used in remote, unattended systems, such as a microprocessor's controlling remote communications switches or repeaters. If parameters need to be changed, the remote switching processor would be telephoned and new parameters transmitted to the E^2PROMs in the system. This application exploits the byte erase feature of the 2816. Only those E^2 locations containing parameters to be changed need be rewritten.

Automatic Receiver — Automatic Transmitter

Fully automatic systems are useful when it is desirable to eliminate the need for a human operator. Here an auto-dial modem is used (previously discussed automatic systems use auto-answer modems). A central computer could be requested to call many remote units to automatically implement program or data update in E^2 memory without human intervention.

To provide an example of one of the four configurations described above, consider a manual receiver-automatic transmitter system. Because the hardware elements of an automatic transmitter are the same as those of an automatic receiver, by considering one example system, all four configurations can be described. With the example that will be discussed, the human operator is on the receiving end and initiates transmission by dialing the transmitter and placing a telephone receiver in an acoustic coupler cradle. The transmitter answers the telephone and transmits data to the receiver which eventually is loaded into E^2PROMs.

RECEIVER

A block diagram of the receiver system is shown in Figure 1. Three elements are of interest here: the modem and modem interface, the receiver CPU and associated software, and the 2816 and E^2 controller.

The receiver CPU is connected to a simple modem which converts serial binary data into acoustical tones. The standard Bell 103 modem or equivalent provides a host system with serial input/output data and various

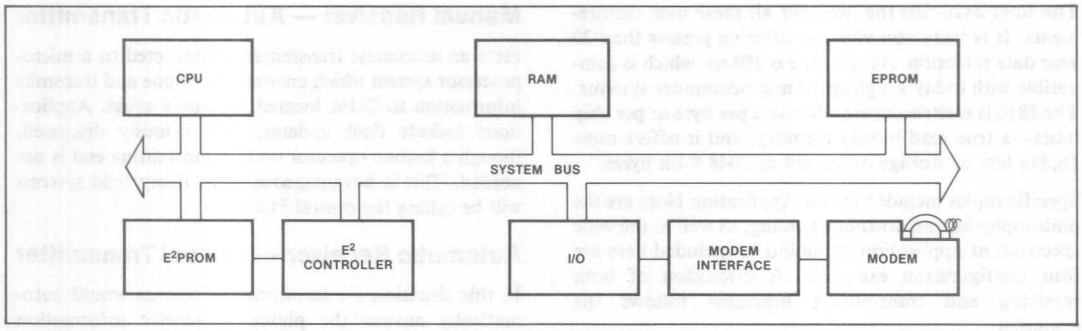
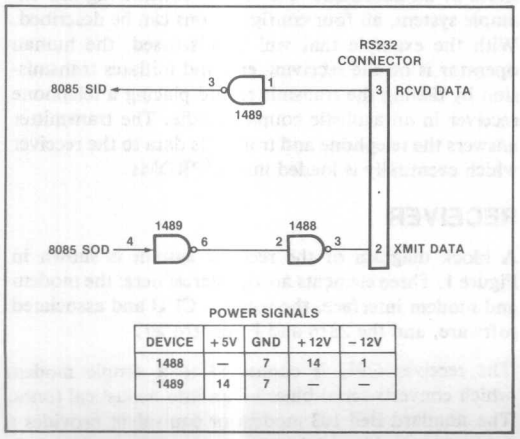


Figure 1. Typical MPU System With E²PROM Memory and Acoustic Coupler

status indicators (such as "carrier detect" which is active when a remote modem carrier signal is detected). The hardware required is minimal since a standard modem can be readily purchased. An RS232 interface is needed to interface 5V TTL signals from a CPU I/O port (or serial data line) to the ±12V RS232 compatible signals of the modem. The rest of the downline load operation is handled in software.

Figure 2 shows a simple modem interface. The MC1489 converts RS232 levels to TTL levels, while the MC1488 converts TTL signals to RS232. In the circuit shown, serial data I/O lines can be passed directly to a UART (Universal Asynchronous Receiver/Transmitter) for serial-parallel data conversion. Another option is to perform the serial-parallel conversion in software. If an 8085 processor is used, the serial I/O lines can be connected to the 8085 SOD and SID ports. The software required is also simple. The receiving CPU only needs to receive data bytes (possibly after a transmitter identification message is received) and program the E²PROM.



DEVICE	+5V	GND	+12V	−12V
1488	—	7	14	1
1489	14	7	—	—

Figure 2. A Simple Modem Interface

Figure 3 contains a flow chart outlining the process of receiving data. The processor first transmits an identifier message, then looks for a return identification message sent from the remote transmitter. This latter message may consist of a sequence of binary or ASCII data detailing the location of the transmitter, date and time of transmission, the number of bytes to be transmitted, the address in E²PROM of where data is to be located, etc. Next, the processor receives a data byte

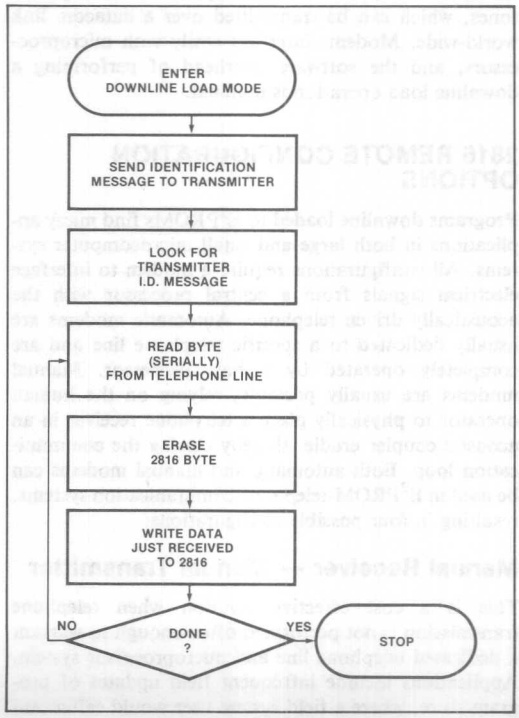


Figure 3. Receiver Software

which may be immediately programmed into the 2816 or saved temporarily in RAM. If serial-to-parallel data conversion is performed by software, data received must be saved in RAM. The 2816 cannot be programmed as each byte is received, since the processor must devote most of its time to receiving data bits and converting them to parallel form. However, if a UART circuit is used to perform data conversion in hardware, data bytes may be saved in E^2 memory as soon as they are received.

To illustrate this, assume data is transmitted at 300 baud (300 bits per second). Assuming each character consists of 1 start bit, 8 data bits, 1 parity bit, and 1 stop bit, then there are 11 bits per character so a character will be received every 36.7 msec. Between every character a 2816 byte must be erased (10 ms) and written (10 ms). Thus we spend 20 ms out of the 36.7 ms we have available during programming, while 16.7 ms of free time is left until the next byte is received.

The final consideration in the downline load receiver is a 2816 controller circuit. (AP102 describes several different controller configurations.) Controller I is convenient to use here. Figure 4 shows a block diagram of the circuit, while Figure 5 contains the circuit diagram. The read operation for the interface is identical to that for EPROMs. To read data, $\overline{CE}$ and $\overline{OE}$ are taken low after addresses are set up.

To write to the 2816, the host processor simply writes to memory. The controller circuit pulls the processor "ready" line low, stalling the CPU and stabilizing addresses and data for the 10 ms write interval while V_{PP} is active. The controller makes the 2816 resemble a slow write RAM except for the necessity of byte erase prior to writing.

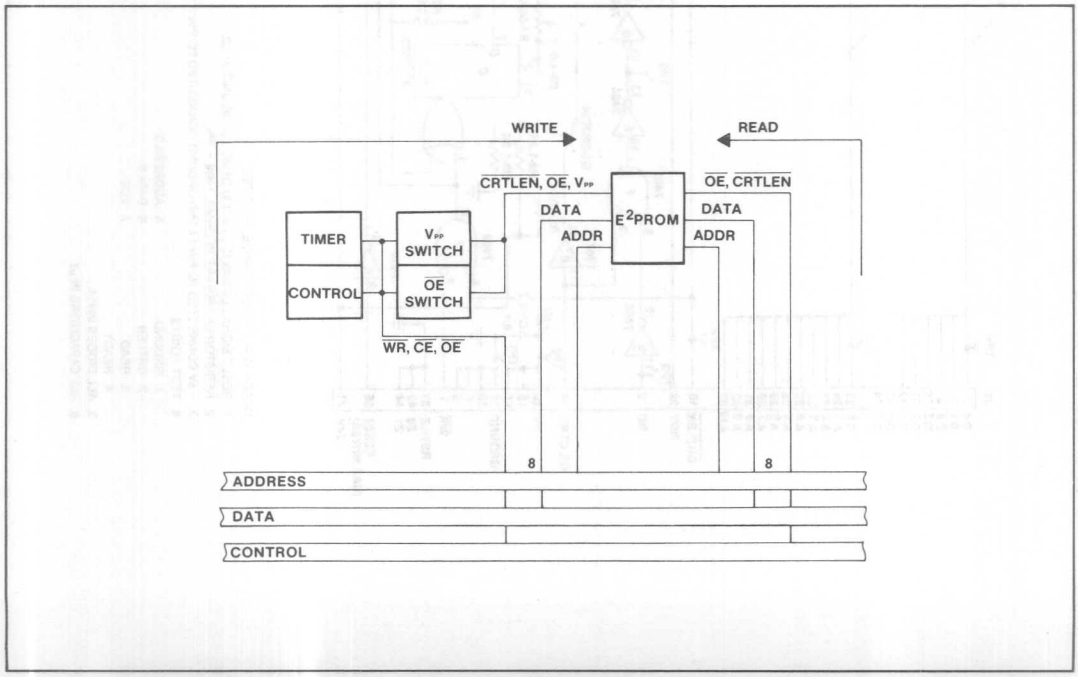
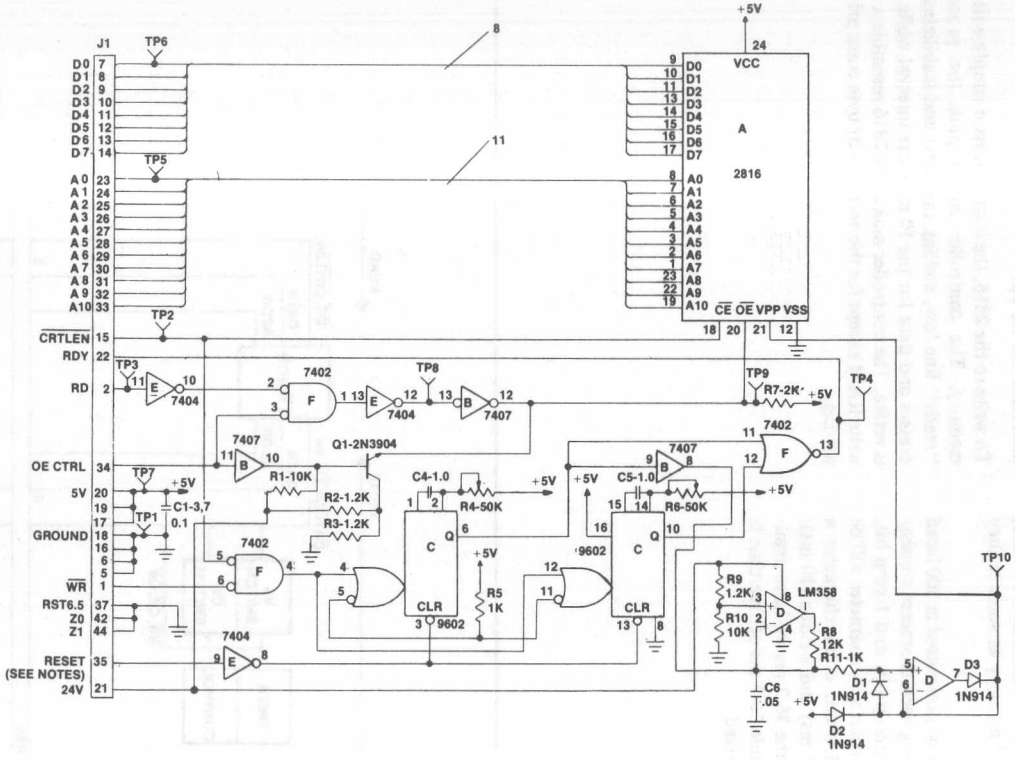


Figure 4. 2816 Controller Block Diagram

AFN-01913A



NOTES: (UNLESS OTHERWISE SPECIFIED)

1. RESET SIGNAL ORIGIN IS SYSTEM DEMONSTOR UNIT J1-22.
2. RESISTOR VALUES ARE IN OHMS, 1/4W, ± 5%.
3. +5V CONNECTED TO PIN 14 AND GROUND CONNECTED TO PIN 7 ON INTEGRATED CIRCUITS.
4. TEST POINTS
   1. GROUND
   2. CRTLEN
   3. READ
   4. READY
   5. ADDRESS 0
   6. DATA 0
   7. VCC
   8. OE CONTROL-READ
   9. RD
   10. VPP
5. ALL DIODES IN914.
6. ALL CAPACITORS IN μf.

Figure 5. E²DEMO Controller I

TRANSMITTER

The transmitter consists of a dedicated microcomputer connected to an auto-answer modem which in turn is attached to a telephone line. The transmit computer software, loops, waiting for an incoming call. When a call is received the modem is signaled to answer the telephone. Information, in the form of data bytes, is received and transmitted in the same fashion as is done on the receiving end. Essentially, all the base station must do is look for a remote processor identification message, send its own identification message, transmit data serially, and hang up the telephone. Additional features may also be implemented such as keeping a log of all calls received, their origins, etc.

Figure 6 contains a block diagram of a base station system. An 8085 processor is used, with an additional 512 bytes of RAM and 4K bytes of EPROM. A modem interface is shown, in addition to a keypad and display for local user operation, and a real-time clock for logging date and time information.

The EPROM memory contains program store and transmit information; i.e., the data that is to be transmitted to remote processor sites. Note that the transmit data EPROM could be replaced by an E^2 device to allow for frequent changes in transmission data without requiring the physical replacement of the transmit data store. RAM is used to save logging information, temporary program data, and a character input buffer which is used to store received characters when looking for a specific message.

The keypad/display module enables a local base station operator to interrogate the base station and reset date or time, access a call log, etc. The clock module is used to keep track of current date and time. Such data may be transmitted to remote processors, or may be used locally as a part of the information logged pertaining to each call received.

A modem interface is very similar to the receiver modem circuit shown in Figure 2. Figure 7 contains a circuit diagram of an auto-answer modem interface. The circuit provides all signals as that of Figure 2, but additionally converts the "Data Terminal Ready" signal and the "Ring Indicator" signal. "Data Terminal Ready" is provided by the host processor and tells the modem when to answer and hang-up the phone line. "Ring Indicator" is active when the phone line is ringing, and is used here to interrupt the processor.

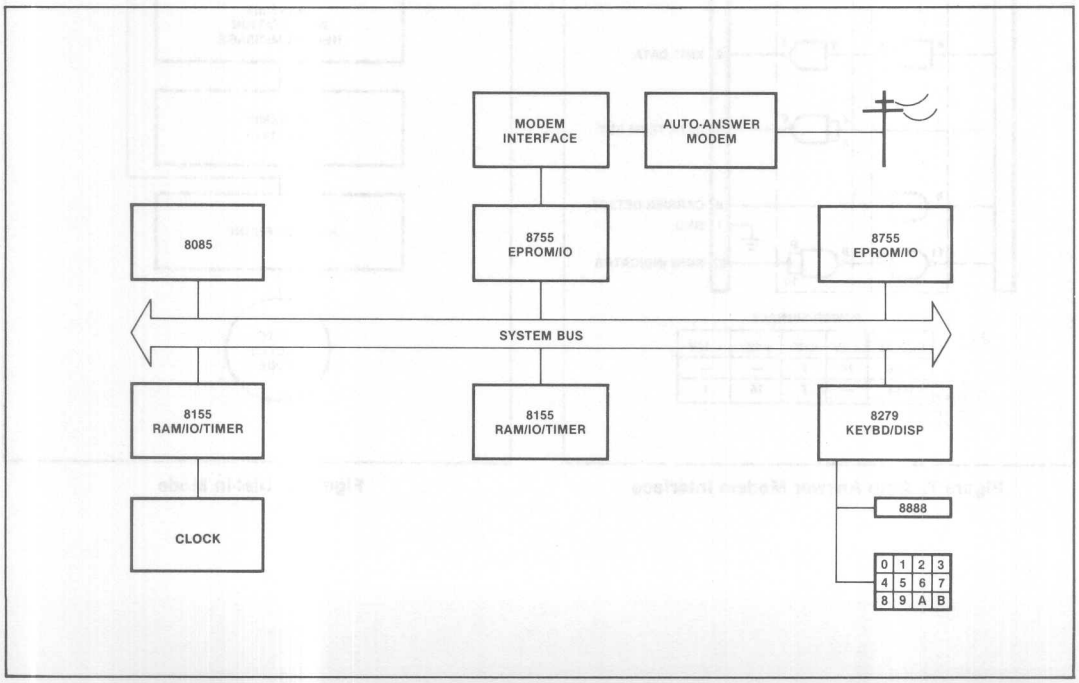
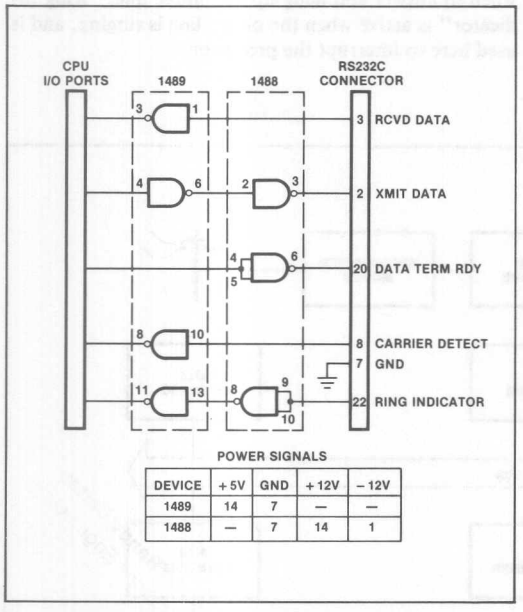


Figure 6. Base Station Block Diagram

Special Products Division Applications Engineering has constructed a base station similar to the one described here. It is used to transmit information to remote 2816s for demonstration purposes. In this unit, software consists of three operating modes:

- **Inactive Mode** is the default. The processor displays the time of day while waiting to enter one of the two modes described below.
- **Dial-In Mode** is entered whenever a call is received. A flow chart of Dial-In Mode software is shown in Figure 8. The processor answers the line, looks for a remote processor identification message, and transmits its own identification header, followed by a text data to be loaded in E^2PROM memory. The telephone is hung up as soon as transmission is completed, and inactive mode is entered.
- **Local User Mode** contains software to allow a local user to reset implemented via the local keypad/display.



POWER SIGNALS

DEVICE	+5V	GND	+12V	-12V
1489	14	7	—	—
1488	—	7	14	1

Figure 7. Auto Answer Modem Interface

CONCLUSION

Remote software changes—that's where 2816 is key. In this application note we've shown the costs involved in field software changes. The 2816 can eliminate field service and maintenance costs involved with software and constant changes. It can do this simply and cheaply through remote data links. Also discussed were typical circuit diagrams and system implementation. The bottom line is that 2816 can eliminate service costs in today's microprocessor systems.

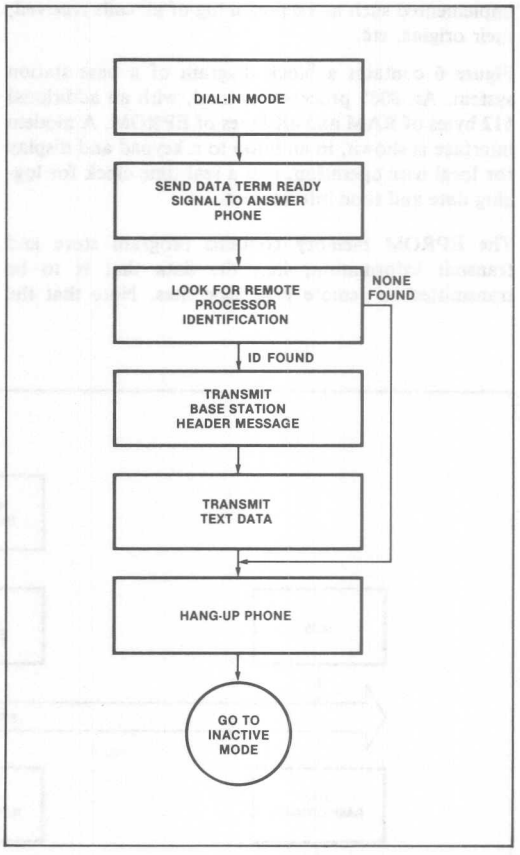


Figure 8. Dial-In Mode